

# Hire the best freelancers for any job, online.

- World's largest freelance marketplace
- Any job you can possibly think of
- Save up to 90% & get quotes for free
- Pay only when you're 100% happy

[Hire a Freelancer](#)[Earn Money Freelancing](#)

# Log In to Freelancer

Email

Password

**Log In**

[Forgot Password?](#)

Don't have an account? [Sign Up](#)

# Create an Account

I want to:



Employee



Freelancer

First Name

Last Name

Email

Password

**Sign Up**

Already have an account? [Log In](#)

# Hire the best freelancers for any job, online.

- World's largest freelance marketplace
- Any job you can possibly think of
- Save up to 90% & get quotes for free
- Pay only when you're 100% happy

[Hire a Freelancer](#)[Earn Money Freelancing](#)

# Describe Your Project

Project Title

e.g., Build a responsive website

Description

Tell us more about your project...

Required Skills

e.g., HTML, CSS, JavaScript

Budget (\$)

e.g., 500

**Post Your Project**

[Back to Home](#)

# Hire Top Talent

Browse our categories to find the perfect freelancer for your project.



## By skill

Looking for a freelancer with a specific skill? Start here.



## By location

Search for freelancers based on their location and timezone.



## By category

Find freelancers that suit a certain project category.



Graphic designers

**Graphic designers**



Website designers

**Website designers**



Mobile app developers

**Mobile app developers**



Software developers

**Software developers**



3D artists

**3D artists**



Illustration

**Illustration**

[Back to Home](#)

# Find Your Next Project

Search for projects based on the skills you have.

Enter skills like 'PHP', 'Design', etc.

Search

## E-commerce

\$300.00

E-commerce, or electronic commerce, involves buying and selling goods and services over the internet. It encompasses all online transactions, from shopping on websites to internet banking and online auctions

html

css

js

nodejs

express

Apply

## E-commerce

\$400.00

E-commerce, or electronic commerce, involves buying and selling goods and services over the internet. It encompasses all online transactions, from shopping on websites to internet banking and online auctions

html

css

js

nodejs

express

Apply

## E-commerce

\$200.00

E-commerce, or electronic commerce, refers to the buying and selling of goods and services over the internet. It encompasses a wide range of online activities, from retail purchases to online auctions and financial transactions. Essentially, it's any business transaction conducted electronically.

java

springboot

html css

View Bids



## E-commerce

\$400.00

E-commerce, or electronic commerce, involves buying and selling goods and services over the internet. It encompasses all online transactions, from shopping on websites to internet banking and online auctions

html

css

js

nodejs

express

Apply

## E-commerce

\$200.00

E-commerce, or electronic commerce, refers to the buying and selling of goods and services over the internet. It encompasses a wide range of online activities, from retail purchases to online auctions and financial transactions. Essentially, it's any business transaction conducted electronically.

java

springboot

html css

View Bids

```
res.status(500).json({ error: 'Failed to assign project or send email' });  
} finally {  
  connection.release();  
}  
});
```

```
app.get('/api/notifications', async (req, res) => {  
  const { userEmail } = req.query;  
  if (!userEmail) return res.status(400).json({ msg: 'User email is required' });  
  try {  
    const sql = 'SELECT * FROM notifications WHERE freelancerEmail = ? ORDER BY createdAt DESC';  
    const [notifications] = await db.query(sql, [userEmail]);  
    res.json(notifications);  
  } catch (err) {  
    console.error("Get Notifications Error:", err);  
    res.status(500).json({ error: err.message });  
  }  
});
```

```
app.get('/api/freelancers/categories', async (req, res) => {  
  try {  
    const [categories] = await db.query('SELECT DISTINCT category, categoryId, img FROM freelancers');  
    res.json(categories.map(c => ({ id: c.categoryId, name: c.category, img: c.img })));  
  } catch (err) {  
    console.error("Get Categories Error:", err);  
    res.status(500).json({ error: err.message });  
  }  
});
```

```
app.get('/api/freelancers', async (req, res) => {  
  try {  
    const [freelancers] = await db.query('SELECT * FROM freelancers WHERE categoryId = ?', [req.query.category]);  
    res.json(freelancers);  
  } catch (err) {  
    console.error("Get Freelancers Error:", err);  
    res.status(500).json({ error: err.message });  
  }  
});
```

```
1 require('dotenv').config();
2 const express = require('express');
3 const path = require('path');
4 const cors = require('cors');
5 const db = require('config/database');
6 const mailer = require('config/mailer');
7
8 const app = express();
9 const PORT = process.env.PORT || 5000;
10
11 const startServer = () => {
12   app.listen(PORT, () => {
13     console.log(`Server running on http://localhost:${PORT}`);
14   });
15 };
16
17 app.use(cors());
18 app.use(express.json());
19 app.use(express.static(path.join(__dirname, 'public')));
20 app.set('trust proxy', true);
21
22 app.post('/api/auth/signup', async (req, res) => {
23   const { firstName, lastName, email, password, role } = req.body;
24   try {
25     if (!role || (role !== 'client' && role !== 'freelancer')) {
26       return res.status(400).json({ msg: 'A valid role is required.' });
27     }
28     const [existingUsers] = await db.query('SELECT email FROM users WHERE email = ?', [email]);
29     if (existingUsers.length > 0) {
30       return res.status(400).json({ msg: 'User already exists' });
31     }
32     const sql = 'INSERT INTO users (firstName, lastName, email, password, role) VALUES (?, ?, ?, ?, ?)';
33     const [result] = await db.query(sql, [firstName, lastName, email, password, role]);
34     res.status(201).json({ msg: 'User created', userId: result.insertId });
35   } catch (err) {
36     console.error("Signup Error:", err);
37     res.status(500).json({ error: err.message });
38   }
39 });
```

```

    }
  });

app.post('/api/auth/login', async (req, res) => {
  const { email, password } = req.body;
  try {
    const [users] = await db.query('SELECT * FROM users WHERE email = ?', [email]);
    if (users.length === 0 || users[0].password !== password) {
      return res.status(400).json({ msg: 'Invalid credentials' });
    }
    const user = users[0];
    const logSql = 'INSERT INTO login_histories (userId, email, ipAddress, userAgent) VALUES (?, ?, ?, ?)';
    await db.query(logSql, [user.id, user.email, req.ip, req.get('User-Agent')]);
    res.json(user);
  } catch (err) {
    console.error("Login Error:", err);
    res.status(500).json({ error: err.message });
  }
});

```

```

app.post('/a const title: any' (req, res) => {
  const { title, description, skills, amount, clientEmail } = req.body;
  try {
    const [users] = await db.query('SELECT role FROM users WHERE email = ?', [clientEmail]);
    if (users.length === 0 || users[0].role !== 'client') {
      return res.status(403).json({ msg: 'Forbidden: Only clients can post projects.' });
    }
    const sql = 'INSERT INTO projects (title, description, skills, amount, clientEmail) VALUES (?, ?, ?, ?, ?)';
    await db.query(sql, [title, description, JSON.stringify(skills), amount, clientEmail]);
    res.status(201).json({ msg: 'Project created' });
  } catch (err) {
    console.error("Post Project Error:", err);
    res.status(500).json({ error: err.message });
  }
});

```

```

app.get('/api/projects', async (req, res) => {
  try {
    let sql = 'SELECT * FROM projects ORDER BY createdAt DESC';
    let params = [];
    if (req.query.skill) {
      sql = 'SELECT * FROM projects WHERE JSON_CONTAINS(skills, ?) ORDER BY createdAt DESC';
      params.push(JSON.stringify(req.query.skill.toLowerCase()));
    }
    if (req.query.clientEmail) {
      sql = 'SELECT * FROM projects WHERE clientEmail = ? ORDER BY createdAt DESC';
      params = [req.query.clientEmail];
    }
    const [projects] = await db.query(sql, params);
    res.json(projects);
  } catch (err) {
    console.error("Get Projects Error:", err);
    res.status(500).json({ error: err.message });
  }
});

app.post('/api/projects/:id/apply', async (req, res) => {
  const { freelancerEmail } = req.body;
  try {
    const [users] = await db.query('SELECT role FROM users WHERE email = ?', [freelancerEmail]);
    if (users.length === 0 || users[0].role !== 'freelancer') {
      return res.status(403).json({ msg: 'Forbidden: Only freelancers can apply for projects.' });
    }
    const sql = 'UPDATE projects SET bids = JSON_ARRAY_APPEND(IFNULL(bids, JSON_ARRAY()), "$", CAST(? AS JSON)) WHERE id = ?';
    await db.query(sql, [JSON.stringify(req.body), req.params.id]);
    res.status(200).json({ msg: 'Application submitted' });
  } catch (err) {
    console.error("Apply to Project Error:", err);
    res.status(500).json({ error: err.message });
  }
});

```

```

app.put('/api/projects/:id/assign', async (req, res) => {
  const projectId = req.params.id;
  const { freelancerEmail } = req.body;
  const connection = await db.getConnection();
  try {
    await connection.beginTransaction();
    await connection.query('UPDATE projects SET status = ?, assignedTo = ? WHERE id = ?', ['assigned', freelancerEmail, projectId]);
    const [projects] = await connection.query('SELECT title, description, clientEmail FROM projects WHERE id = ?', [projectId]);
    const project = projects[0];
    const googleFormLink = 'https://docs.google.com/forms/d/e/your-real-form-id-here/viewform';
    const message = `Congratulations! You have been assigned the project "${project.title}" by ${project.clientEmail}. Please complete the assign
    const fullMessage = `${message}\n\nForm Link: ${googleFormLink}`;
    const notificationSql = 'INSERT INTO notifications (freelancerEmail, clientEmail, projectId, projectName, message) VALUES (?, ?, ?, ?, ?)';
    await connection.query(notificationSql, [freelancerEmail, project.clientEmail, projectId, project.title, fullMessage]);
    const mailOptions = {
      from: `"Freelancer Platform" <${process.env.EMAIL_USER}>`,
      to: freelancerEmail,
      subject: `You've been assigned a new project: "${project.title}"`,
      html: `
        <p>Hello,</p>
        <p>Congratulations! You have been assigned the project "<b>${project.title}</b>" by the client (<b>${project.clientEmail}</b>).</p>
        <h3>Project Description:</h3>
        <p>${project.description}</p>
        <p>Please complete the project details form at the following link to provide your GitHub repository and payment information:</p>
        <a href="${googleFormLink}" style="background-color: #00B2E3; color: white; padding: 10px 20px; text-decoration: none; border-radius: 5px;>
        <p><br>Thank you,<br>The Freelancer Platform Team</p>
      `;
    };
    await mailer.sendMail(mailOptions);
    await connection.commit();
    res.status(200).json({ msg: 'Project assigned and email notification sent.' });
  } catch (err) {
    await connection.rollback();
    console.error("Assign Project Error:", err);
    res.status(500).json({ error: 'Failed to assign project or send email.' });
  }
});

```

```
res.status(500).json({ error: 'Failed to assign project or send email' });  
} finally {  
  connection.release();  
}  
});
```

```
app.get('/api/notifications', async (req, res) => {  
  const { userEmail } = req.query;  
  if (!userEmail) return res.status(400).json({ msg: 'User email is required' });  
  try {  
    const sql = 'SELECT * FROM notifications WHERE freelancerEmail = ? ORDER BY createdAt DESC';  
    const [notifications] = await db.query(sql, [userEmail]);  
    res.json(notifications);  
  } catch (err) {  
    console.error("Get Notifications Error:", err);  
    res.status(500).json({ error: err.message });  
  }  
});
```

```
app.get('/api/freelancers/categories', async (req, res) => {  
  try {  
    const [categories] = await db.query('SELECT DISTINCT category, categoryId, img FROM freelancers');  
    res.json(categories.map(c => ({ id: c.categoryId, name: c.category, img: c.img })));  
  } catch (err) {  
    console.error("Get Categories Error:", err);  
    res.status(500).json({ error: err.message });  
  }  
});
```

```
app.get('/api/freelancers', async (req, res) => {  
  try {  
    const [freelancers] = await db.query('SELECT * FROM freelancers WHERE categoryId = ?', [req.query.category]);  
    res.json(freelancers);  
  } catch (err) {  
    console.error("Get Freelancers Error:", err);  
    res.status(500).json({ error: err.message });  
  }  
});
```

```
const [freelancers] = await db.query('SELECT * FROM freelancers WHERE categoryId = ?', [req.query.category]);
res.json(freelancers);
} catch (err) {
  console.error("Get Freelancers Error:", err);
  res.status(500).json({ error: err.message });
}
});
```

```
app.get('/api/freelancers/:id', async (req, res) => {
  try {
    const [freelancers] = await db.query('SELECT * FROM freelancers WHERE id = ?', [req.params.id]);
    if (freelancers.length === 0) {
      return res.status(404).json({ msg: 'Freelancer not found' });
    }
    res.json(freelancers[0]);
  } catch (err) {
    console.error("Get Freelancer by ID Error:", err);
    res.status(500).json({ error: err.message });
  }
});
```

```
db.getConnection()
  .then(connection => {
    console.log('Database connection verified. Starting server...');
    connection.release();
    startServer();
  })
  .catch(err => {
    console.error('FATAL: Could not connect to the database. Server will not start.');
```

```
    console.error(err);
    process.exit(1);
  });
```